Design a Win-Win Strategy That Is Fair to Both Service Providers and Tasks When Rejection is Not an Option

Yohai Trabelsi[1] , **Pan Xu**[2] , **Sarit Kraus**[1]
[1]Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
[2]New Jersey Institute of Technology, Newark, NJ, USA
yohai.trabelsi@gmail.com, pxu@njit.edu, sarit@cs.biu.ac.il

Abstract

Assigning tasks to service providers is a frequent procedure across various applications. Often the tasks arrive dynamically while the service providers remain static. Preventing task rejection caused by service provider overload is of utmost significance. To ensure a positive experience in relevant applications for both service providers and tasks, fairness must be considered. To address the issue, we model the problem as an online matching within a bipartite graph and tackle two minimax problems: one focuses on minimizing the highest waiting time of a task, while the other aims to minimize the highest workload of a service provider. We show that the second problem can be expressed as a linear program and thus solved efficiently while maintaining a reasonable approximation to the objective of the first problem. We developed novel methods that utilize the two minimax problems. We conducted extensive simulation experiments using real data and demonstrated that our novel heuristics, based on the linear program, performed remarkably well.

1 Introduction

In resource allocation, numerous problems can be represented as online matching in bipartite graphs. One side of the graph comprises service providers (interchangeably called workers in this paper), while the other consists of allocated task types. The graph's edges indicate the qualifications of service providers to perform tasks of specific types.

In online matching problems, a common scenario involves one dynamic side and one static side. This dynamic-static setup finds application in various contexts, such as matching riders(dynamic) to drivers(static) [Dickerson *et al.*, 2021], connecting search queries(dynamic) to advertisers in sponsored search(static) [Delong *et al.*, 2022], and facilitating the teleoperation of autonomous vehicles (AVs) [Ackerman Viden *et al.*, 2023]. The primary objective in these problems is to optimize some criteria from the perspective of the allocator.

Some other works are dedicated to optimizing allocation fairness. For example, in the domain of ride-sourcing, a method to achieve allocation fairness was proposed in [Lesmana *et al.*, 2019]. Additionally, certain studies address cases where fairness should be maintained for both online tasks and offline workers [Esmaeili *et al.*, 2023].

Our work is motivated by the teleoperation of AVs that has garnered increasing attention recently (e.g., [Zhang, 2020; Ackerman Viden *et al.*, 2023; Tener and Lanir, 2022]). The primary role of teleoperation is to aid AVs by intervening in challenging driving situations[1]. Ensuring a fair allocation of teleoperators to driving tasks is crucial for enhancing the satisfaction of both teleoperators and AVs' users. Particularly, if certain intervention requests have significantly longer waiting times or if some teleoperators are disproportionately busier than others, such imbalances can lead to dissatisfaction among those affected. In addition, as a person in the vehicle is awaiting the teleoperator's intervention, a rejection of a request is unacceptable. Another property of this application is that the teleoperators (workers) are reusable, which means they are ready to perform a new intervention request (task) once they finish a previously allocated request.

We model the problem as online matching in a bipartite graph and propose several approaches to optimize fairness for both the tasks (e.g., intervention requests) and the workers (e.g., teleoperators) involved in the process. Our notion of fairness is aligned with Rawls' theory of justice [Rawls, 1999].

We introduce two minimax problems within the given context. The first concerns fairness regarding tasks relative to waiting times, while the second focuses on Rawlsian fairness for service providers based on their workload. In both scenarios, task rejection is not permissible. We demonstrate that the second problem can be efficiently formulated as a linear problem. Notably, the solution to the second problem mirrors the first when task durations from each worker conform to the same distribution. In cases where this isn't true, we show that the second problem's solution approximates the first problem's solution, supported by a provable approximation ratio. Our study concludes with extensive simulations that underscore the efficacy of these minimax problems. Furthermore, we devise innovative heuristics that leverage the minimax solutions. These heuristics enhance task fairness while preserving favorable outcomes for worker fairness.

Our main contributions are: (1) We propose two models to promote fairness among tasks and workers. (2) We

[1]As mentioned in [Tener and Lanir, 2022], the AVs will need this intervention, at least in the near future.

present an LP-based algorithmic framework, which can exactly solve fairness maximization among workers and approximately among tasks, and we provide a tight approximation bound. (3) We empirically implement and compare different methods, including several baselines, on datasets involving the teleoperation of AVs.

1.1 Related Work

In this section, we describe previous works about fair allocation and allocation with delays. Notably, to our knowledge, our work distinguishes itself by being the first to consider fairness and allocation delays together.

Fair Allocation Some studies address fair allocation, focusing on only one side of the graph, as seen in [Ma *et al.*, 2020]. Although their fairness approach resembles ours, it pertains solely to one side of the graph, which falls short of our requirements. Other research, like [Patro *et al.*, 2020], deals with fairness in recommendation systems. However, the fairness objectives in recommendation systems significantly differ from those in task allocation contexts. Practical solutions for enhancing fairness for both service providers and tasks are explored in works such as [Zhou *et al.*, 2023]. Regrettably, this branch of research lacks theoretical performance bounds for their solutions. The fairness principles in [Esmaeili *et al.*, 2023] closely align with ours. They consider both workers (offline side) and tasks (online side), embracing Rawlsian welfare [Rawls, 1958]. Nonetheless, task rejection is permissible in their scenario if workers are unavailable.

Allocation with Delayed Assignments The original online matching problem was introduced in [Karp *et al.*, 1990], where static nodes (workers) are instantly paired with dynamic nodes (tasks) upon arrival. However, real scenarios often lack immediate worker availability for tasks, prompting consideration for task execution delays over outright rejection. Numerous works tackle resource allocation with potential task delays. However, many of these approaches (e.g., [Righter, 1987; Li *et al.*, 2023]) prioritize utility maximization without factoring in task wait times or worker workload. Some leverage reinforcement learning for such issues yet often make batch decisions, leading to suboptimal outcomes. Moreover, theoretical guarantees are frequently absent. An LP-based method for delayed allocations is presented in [Ackerman Viden *et al.*, 2023], optimizing a complex utility function that accounts for task waiting times but overlooks worker workload.

Another pertinent domain involves queue admission control systems with multiple classes. Here, diverse customer types (tasks) arrive dynamically, and a decision-maker determines which task to accept, as demonstrated in [Rigter *et al.*, 2022]. However, several studies in this realm do not distinguish between workers, while others permit task rejection. To our knowledge, the problem of two-sided fair allocation when task rejection is not allowed has not yet been addressed.

2 Preliminaries

Suppose we use a bipartite graph $G = (I, J, E)$ to model the worker-task network, where I denotes the set of offline workers (*e.g.,* teleoperators), J the set of types of tasks, and an edge $e = (i, j)$ indicates the feasibility of worker i to serve the task (of type) j. Note that at certain points within this paper, we abuse the notation by referring to j as a task instead of a task type. We also abuse the notation by referring to an edge $e = (i, j)$ as (ij). Tasks of type $j \in J$ arrive following an independent Poisson process of rate $\lambda_j > 0$. For each edge $e = (i, j) \in E$, we assume it takes worker i an exponentially distributed service time[2] of rate $\mu_{ij} > 0$ to complete a task of type j (*i.e.,* with mean of $1/\mu_{ij}$)[3]. For each worker i and task j, let $\mathcal{N}_i \subseteq J$ and $\mathcal{N}_j \subseteq I$ denote the set of neighbors of i and j in the graph G. **The assigning rule** is as follows. Upon the arrival of a task of type j, we (as the central coordinator) have to assign it to a feasible worker $i \in \mathcal{N}_j$ immediately: if i is free (or available) at that time, then i will serve j right away; otherwise, j will join the virtual queue of i and it will stay there until being served by i.

Table 1: A glossary of notations throughout this paper.

G	Input network graph $G = (I, J, E)$.
I (J)	Set of worker (task) types.
$\mathcal{N}_i$ ($\mathcal{N}_j$)	Set of neighbors of i (j).
$i \sim j$ ($j \sim i$)	Equivalent to $i \in \mathcal{N}_j$ ($j \in \mathcal{N}_i$).
λ_j	Arrival rate of task type $j \in J$.
λ_i	Arrival rate on worker $i \in I$.
$\mathrm{Exp}(\mu)$	Exponential distribution of rate $\mu > 0$.
$\mathrm{Exp}(\mu_{ij})$	Service time taken by worker i to service j.
$\rho_i \in [0, 1]$	Workload of worker $i \in I$.
w_j	Expected (absolute) waiting time of j; see Eqn. (4).
$\bar{w}_j$	Expected (relative) waiting time of j; see Eqn. (5).
$\kappa \geq 1$	$\max_{i \in I} \left(\max_{j \sim i, j' \sim i} \mu_{ij}/\mu_{ij'} \right)$.

2.1 Allocation Policy and Related Concepts

Consider an allocation policy $\pi(\mathbf{x})$ (possibly randomized), characterized as a vector $\mathbf{x} = \{x_{ij} | (ij) \in E\}$, where $x_{ij} \in [0, 1]$ denotes the percentage of task (of type) j assigned to and served by worker i. In the following, we discuss a few important properties and concepts related to $\pi(\mathbf{x})$. Let Q_i be the virtual queue maintained by worker $i \in I$.

Arrival rate on Q_i, denoted by λ_i. Observe that $\mathbf{x} = (x_{ij})$ can be viewed alternatively as the probability that π assigns each arriving j to i. Thus, we claim that Q_i admits a Poisson arrival process of rate $\lambda_i := \sum_{j \in \mathcal{N}_i} \lambda_j \cdot x_{ij}$. By the property of the Poisson process (See section 2.3.2 at [Gallager, 2011]), conditioning on the arrival of task (of type) $\bar{j} \in J$ on i, we claim that $\Pr[\bar{j} = j] = x_{ij} \cdot \lambda_j / \lambda_i$ for each $j \in \mathcal{N}_i$.

Service time on Q_i, denoted by $\mathcal{S}_i$. The analysis above shows that the task joining Q_i is of type $j \in \mathcal{N}_i$ with probability equal to $x_{ij} \cdot \lambda_j / \lambda_i$. Thus, the overall service time $\mathcal{S}_i = \sum_{j \in \mathcal{N}_i} \chi_{ij} \cdot \mathrm{Exp}(\mu_{ij})$, where $\chi_{ij} = 1$ indicates that the task joining i is of type j with $\mathsf{E}[\chi_{ij}] = x_{ij} \cdot \lambda_j / \lambda_i$, and $\mathrm{Exp}(\mu_{ij})$ represents the

[2]This assumption is justified in [Devore, 2008]. Note that the theoretical analysis does not depend on it. We could use any distribution if the mean and the variance of service time are known.

[3]Note that the assumption does not necessarily suggest the most likely outcome is for tasks to be finished in an extremely short time. Consider a task type with an exponentially distributed service time of rate μ, denoted as $X = \mathrm{Exp}(\mu)$. We observe that for any given threshold $a > 0$, $\Pr[X \geq a] = e^{-\mu a}$, which can be close to one when μ is small.

exponentially distributed service time of i for j of rate μ_{ij}. Thus, $\mathcal{S}_i$ follows a *hyperexponential distribution* [Gupta and Goyal, 1964] with mean equal to

$$s_i := \mathsf{E}[\mathcal{S}_i] = \sum_{j \in \mathcal{N}_i} (x_{ij}\lambda_j)/(\lambda_i \mu_{ij}). \tag{1}$$

Workload of worker i, denoted by ρ_i. By definition,

$$\rho_i := \lambda_i \cdot \mathsf{E}[\mathcal{S}_i] = \sum_{j \in \mathcal{N}_i} (x_{ij}\lambda_j)/\mu_{ij}, \tag{2}$$

where ρ_i can be re-interpreted as the probability that the worker i is busy or the proportion of time the worker i is busy averaged over a long period. Note that $\rho_i < 1$ is the key condition ensuring the virtual queue Q_i can enter a stable state. This is also a condition we should impose on every worker $i \in I$ when designing policy $\pi(\mathbf{x})$ since otherwise, i could always stay occupied in the long run (thus, not acceptable to i) and every task j assigned to i could risk an infinitely long waiting time (not acceptable to j).

Waiting time on worker i, denoted by W_i. By the analysis above, we see that the queue Q_i on worker i qualifies as an $M/G/1$ (using the standard Kendall's notation [Kendall, 1953]), which means it admits a Poisson arrival process, a general service time distribution, and a single worker. By the Pollaczek-Khinchin mean formula [Asmussen, 2003],

$$w_i := \mathsf{E}[W_i] = \frac{\lambda_i \mathsf{E}[\mathcal{S}_i^2]}{2(1-\rho_i)} = \frac{\sum_{j \in \mathcal{N}_i} x_{ij}\lambda_j/\mu_{ij}^2}{1 - \sum_{j \in \mathcal{N}_i} x_{ij}\lambda_j/\mu_{ij}}, \tag{3}$$

where the numerator is equal to

$$\begin{aligned}
&\lambda_i \cdot \mathsf{E}[\mathcal{S}_i^2] = \lambda_i \cdot \mathsf{E}\Big[\Big(\sum_{j \in \mathcal{N}_i} \chi_{ij} \cdot \mathrm{Exp}(\mu_{ij})\Big)^2\Big] \\
&= \lambda_i \cdot \mathsf{E}\Big[\sum_{j \in \mathcal{N}_i} \chi_{ij} \cdot \mathrm{Exp}^2(\mu_{ij})\Big] = \lambda_i \cdot \sum_{j \in \mathcal{N}_i} \mathsf{E}\Big[\chi_{ij} \cdot \mathrm{Exp}^2(\mu_{ij})\Big] \\
&= \lambda_i \cdot \sum_{j \in \mathcal{N}_i} (x_{ij}\lambda_j/\lambda_i) \cdot (2/\mu_{ij}^2) = 2 \sum_{j \in \mathcal{N}_i} (x_{ij}\lambda_j/\mu_{ij}^2).
\end{aligned}$$

Absolute and relative waiting time of j, denoted by W_j and $\overline{W}_j$. Recall that under $\pi(\mathbf{x})$, a task j will be assigned to a feasible worker $i \in \mathcal{N}_j$ with probability x_{ij}. Thus, the expected (absolute) waiting time of j should be

$$w_j := \mathsf{E}[W_j] = \sum_{i \in \mathcal{N}_j} x_{ij} \cdot w_i, \tag{4}$$

where w_i is the expected waiting time on queue Q_i, as shown in (3). The *relative* waiting time of j on Q_i is defined as the ratio of waiting time on Q_i to the service time of i for j, which has a mean of $1/\mu_{ij}$. Thus, the expected relative waiting time of j should be

$$\begin{aligned}
\bar{w}_j := \mathsf{E}[\overline{W}_j] &= \sum_{i \in \mathcal{N}_j} x_{ij} \cdot w_i/(1/\mu_{ij}) \\
&= \sum_{i \in \mathcal{N}_j} x_{ij} \cdot \mu_{ij} \cdot \frac{\sum_{\ell \sim i} x_{i\ell}\lambda_\ell/\mu_{i\ell}^2}{1 - \sum_{\ell \sim i} x_{i\ell}\lambda_\ell/\mu_{i\ell}}.
\end{aligned} \tag{5}$$

2.2 Two Fairness-Related Objectives

In this paper, we propose the following two fairness metrics and objectives when optimizing a policy $\pi(\mathbf{x})$.

FAIR-T: Fairness promotion among tasks, denoted by $\min\max_{j \in J} \bar{w}_j$. We quantify the overall fairness among users achieved by policy $\pi(\mathbf{x})$ as the maximum expected *relative* waiting time among all task types, *i.e.,* $\max_{j \in J} \bar{w}_j$. A formula for calculating the relative waiting time is shown in (5). Note that here we choose the relative version instead of the absolute one (*i.e.,* $\max_{j \in J} w_j$) following, for example, the paper [Maister and others, 1984] that asserts that "the more valuable the service, the longer the customer will wait." A compelling example is that:"Special checkout counters were originally provided because customers with only a few items felt resentful at having to wait a long time for what was seen as a simple transaction. Customers with a full cart of groceries were much more inclined to tolerate lines."

FAIR-S: Fairness promotion among workers, denoted by $\min\max_{i \in I} \rho_i$. Recall that for each worker $i \in I$, the workload $\rho_i \in (0, 1)$, as defined in (2), captures the percentage of busy time on worker i. Thus, the maximum workload, i.e., $\max_{i \in I} \rho_i$, reflects the highest degree of being occupied among all workers under policy $\pi(\mathbf{x})$. By opting for minimization of the maximum workload, denoted by $\min\max_{i \in I} \rho_i$, we aim to minimize the occupation time of the most occupied worker as substantially as feasible.

2.3 Two Optimization Programs

Consider an allocation policy $\pi(\mathbf{x})$ parameterized by $\mathbf{x} = (x_{ij})$, where x_{ij} with $(ij) \in E$ denotes the percentage of task of type j assigned to worker i. For ease of notation, we will use $i \sim j$ (and $j \sim i$) to represent $i \in \mathcal{N}_j$ (and $j \in \mathcal{N}_i$) throughout this paper. We formulate **FAIR-T** and **FAIR-S** as minmax programs as follows.

$$(\mathbf{PT}) \min \max_{j \in J} \Big(\bar{w}_j = \sum_{i \sim j} x_{ij} \cdot \mu_{ij} \cdot \frac{\sum_{\ell \sim i} x_{i\ell}\lambda_\ell/\mu_{i\ell}^2}{1 - \sum_{\ell \sim i} x_{i\ell}\lambda_\ell/\mu_{i\ell}}\Big), \tag{6}$$

$$x_j := \sum_{i \sim j} x_{ij} = 1, \ \ \forall j \in J \tag{7}$$

$$\rho_i = \sum_{\ell \sim i} x_{i\ell}\lambda_\ell/\mu_{i\ell} \le 1, \ \ \forall i \in I \tag{8}$$

$$0 \le x_{ij} \le 1, \ \ \forall (ij) \in E. \tag{9}$$

$$(\mathbf{PS}) \ \ \min \ \ \max_i \rho_i, \tag{10}$$

$$x_j := \sum_{i \sim j} x_{ij} = 1, \qquad \forall j \in J \tag{11}$$

$$\rho_i = \sum_{\ell \sim i} x_{i\ell}\lambda_\ell/\mu_{i\ell} \le 1, \ \ \forall i \in I \tag{12}$$

$$0 \le x_{ij} \le 1, \qquad \forall (ij) \in E. \tag{13}$$

We refer to the above programs as **PT** and **PS**, respectively. Let $\mathbf{x}_t^*$ and $\mathbf{x}_s^*$ be optimal solutions to **PT** and **PS**, respectively.

Lemma 1. *$\pi(\mathbf{x}_t^*)$ and $\pi(\mathbf{x}_s^*)$ are optimal policies under **FAIR-T** and **FAIR-S**, respectively.*

Proof. We focus on showcasing the case of **FAIR-T** and the program **PT**. The proof for the other case is similar. Note that the term shown in (6) captures the precise objective we aim to optimize. To prove our claim, we need to demonstrate that all constraints in **PT** hold true for any viable policy of $\pi(\mathbf{x})$. Constraint (7) is reasonable because every policy must assign each incoming task to a feasible worker without rejection, thereby ensuring that the total percentages assigned for each type sum up to one. Constraint (8) is valid as the workload of any worker (i.e., the percentage of busy time) should not exceed one. Constraint (9) holds true since x_{ij} represents the percentage of tasks of type j assigned to worker i. □

Lemma 1 suggests that the optimal policies for **FAIR-T** and **FAIR-S** each can be obtained by solving minmax programs represented by **PT** and **PS** respectively. Note that **PS** can be reformulated as a linear program (LP) by introducing an auxiliary variable ρ and modifying the objective as $\min \rho$, along with additional constraints $\rho \geq \rho_i$ for all $i \in I$. Consequently, we can efficiently solve **PS** and obtain an optimal policy for **FAIR-S**. However, for program **PT**, the objective is non-linear and can be neither convex nor concave even under very special settings, posing a technical challenge for direct optimization; see detailed discussions in the full version of the paper [Trabelsi *et al.*, 2024].

Nevertheless, under certain conditions, **PT** can be effectively and accurately approximated by **PS**, as proven in Theorem 1.

Lemma 2. *The optimal values of **PT** and **PS** each remain invariant if we treat any task type $j \in J$ with an arrival rate of λ_j as k different online types, each having the same set of neighbors as j, with an arrival rate of λ_j/k for any integer k.*

The above lemma suggests that for fairness maximization among either workers under metric **FAIR-S** or tasks under metric **FAIR-T**, we can assume without loss of generality that all tasks take a uniform arrival rate by creating an appropriate number of copies for each task type. In other words, the variation among tasks' arrival rates makes no difference to fairness promotion, compared with the difference among service times. ***In the remaining sections, we assume without loss of generality that $\lambda_j = \lambda$ for all $j \in J$.***

3 The Relation Between the Two Fairness Optimization Problems

Consider a general setting denoted by $\boldsymbol{\mu} := (\mu_{ij})$, where μ_{ij} with $(ij) \in E$ represents the parameter for the exponential distribution of the service time taken by worker i to serve task j. Let $\eta_t(\boldsymbol{\mu}, \mathbf{x})$ denote the objective value of $\mathbf{PT}(\boldsymbol{\mu})$ with respect to the input $\boldsymbol{\mu}$ and a feasible solution $\mathbf{x} = (x_{ij})$. Similarly, $\eta_s(\boldsymbol{\mu}, \mathbf{x})$ denotes the objective value of $\mathbf{PS}(\boldsymbol{\mu})$. When the context is clear, we may omit either the first or second argument for η_t and η_s. For any given input $\boldsymbol{\mu}$, let $\eta_t^*(\boldsymbol{\mu})$ and $\eta_s^*(\boldsymbol{\mu})$ denote the optimal values of $\mathbf{PT}(\boldsymbol{\mu})$ and $\mathbf{PS}(\boldsymbol{\mu})$ respectively.

Theorem 1. *Let $\mathbf{x}_s^*$ be an optimal solution to $\boldsymbol{PS}(\boldsymbol{\mu})$. We have*

$$\eta_t(\boldsymbol{\mu}, \mathbf{x}_s^*) \leq \kappa^3 \left(1 + \left(1 - \frac{1}{\kappa}\right) \cdot \frac{\eta_s^*(\boldsymbol{\mu})}{1 - \eta_s^*(\boldsymbol{\mu})}\right) \cdot \eta_t^*(\boldsymbol{\mu}), \quad (14)$$

where $\kappa = \max_{i \in I} \left(\max_{j \sim i, j' \sim i} \mu_{ij}/\mu_{i,j'}\right) \geq 1$, which captures the maximum pairwise ratio among the expectations of all service time on each given worker.

For a private case of Theorem 1- where $\kappa = 1$ we prove that $\eta_t(\boldsymbol{\mu}, \mathbf{x}_s^*) = \eta_t^*(\boldsymbol{\mu})$ (Theorem 2).

These results serve as the bedrock of the whole proof for Theorem 1(See [Trabelsi *et al.*, 2024] for the whole proof of Theorem 1). Toward the proof of $\kappa = 1$, we first define the following minimax programs and show their equivalence to **PT** and **PS**, respectively, for $\kappa = 1$.

$$(\overline{\mathbf{PT}}) \quad \min \ \max_{j \in J} \left(\bar{w}_j = \sum_{i \sim j} \frac{x_{ij}}{1 - \rho_i} - 1\right), \quad (15)$$

$$\sum_{i \sim j} x_{ij} = 1, \qquad \forall j \in J \quad (16)$$

$$\rho_i = (\lambda/\mu_i) \sum_{j \sim i} x_{ij} \leq 1, \qquad \forall i \in I \quad (17)$$

$$0 \leq x_{ij} \leq 1, \qquad (ij) \in E. \quad (18)$$

$$(\overline{\mathbf{PS}}) \quad \min \ \max_{i \in I} \rho_i, \quad (19)$$

$$\sum_{i \sim j} x_{ij} = 1, \qquad \forall j \in J \quad (20)$$

$$\rho_i = (\lambda/\mu_i) \sum_{j \sim i} x_{ij} \leq 1, \ \forall i \in I \quad (21)$$

$$0 \leq x_{ij} \leq 1, \qquad (ij) \in E. \quad (22)$$

Lemma 3. *For $\kappa = 1$, the programs **PT** and $\overline{\textbf{PT}}$ are equivalent and the programs **PS** and $\overline{\textbf{PS}}$ are also equivalent.*

Proof. Note that $\kappa = 1$ suggests that μ_{ij} takes some uniform value of $\mu_{ij} = \mu_i$ for every $j \sim i$. Recall that $\lambda_j = \lambda$ for all $j \in J$ due to Lemma 2. Under these assumptions, we see that the expressions of ρ_i and $\bar{w}_j$ in (2) and (5) can be simplified as

$$\rho_i = \sum_{\ell \sim i} x_{i\ell} \lambda_\ell / \mu_{i\ell} = (\lambda/\mu_i) \cdot \sum_{\ell \sim i} x_{i\ell},$$

$$\bar{w}_j := \mathsf{E}[\overline{W}_j] = \sum_{i \sim j} x_{ij} \cdot w_i/(1/\mu_{ij})$$

$$= \sum_{i \sim j} \frac{x_{ij} \cdot \mu_i \cdot \sum_{\ell \sim i} x_{i\ell} \lambda_\ell / \mu_i^2}{1 - \sum_{\ell \sim i} x_{i\ell} \lambda_\ell / \mu_i} = \sum_{i \sim j} \frac{x_{ij} \cdot \sum_{\ell \sim i} x_{i\ell} \lambda_\ell / \mu_i}{1 - \sum_{\ell \sim i} x_{i\ell} \lambda_\ell / \mu_i}$$

$$= \sum_{i \sim j} x_{ij} \left(-1 + \frac{1}{1 - \rho_i}\right) = -1 + \sum_{i \sim j} \frac{x_{ij}}{1 - \rho_i},$$

where the equality on the last line is due to $\sum_{i \sim j} x_{ij} = 1$ for every $j \in J$ (no rejection allowed). Substituting lines 17 and 21 with the value of ρ_i and line 15 with the value of $\bar{w}_j$ implies that the programs **PT** and **PS** are equivalent to $\overline{\textbf{PT}}$ and $\overline{\textbf{PS}}$. □

Consider a given setting with $\boldsymbol{\mu} = (\mu_{ij})$ satisfying $\mu_{ij} = \mu_i$ for all $j \sim i$ and $\lambda_j = \lambda$ for all $j \in J$. For ease of notation, we use $\overline{\eta_t^*}$ and $\overline{\eta_s^*}$ to denote optimal values of $\overline{\textbf{PT}}$ and $\overline{\textbf{PS}}$, respectively, with respect to the given setting. By default, we

assume both have feasible solutions.[4] We denote by $\overline{\eta_t(\mathbf{x}_s^*)}$ the value of $\overline{\boldsymbol{PT}}$ on $\mathbf{x}_s^*$ in the given setting.

It is tempting to prove that $\eta_t(\mathbf{x}_s^*) = \eta_t^*$ for $\kappa = 1$ by showing that $\overline{\mathbf{PT}}$ and $\overline{\mathbf{PS}}$ each possess an optimal solution such that $\{\rho_i\}$ all take a uniform value, say ρ. Following this "claim", $\overline{\mathbf{PT}}$ is then reduced to $\min 1/(1-\rho) - 1$ with $\rho = \rho_i$ for all $i \in I$, while $\overline{\mathbf{PS}}$ is reduced to $\min \rho$ with $\rho = \rho_i$ for all $i \in I$. This establishes Theorem 2 since $\min 1/(1-\rho) - 1$ is equivalent to $\min \rho$. The example below disproves this idea, unfortunately.

Example 1. *[$\overline{\mathbf{PT}}$ and $\overline{\mathbf{PS}}$ each possess a unique optimal solution with non-uniform values of $\{\rho_i\}$ and $\{\bar{w}_j\}$.] Consider a graph $G = (I, J, E)$ such that $|I| = m = 2$ and $|J| = n \gg 1$ (A relevant figure is in [Trabelsi* et al.*, 2024]). The input setting is as follows. $\mu_{ij} = \mu$ for all $(ij) \in E$ and $\lambda_j = \lambda$ for all $j \in J$. Let $\phi = \lambda/\mu$ with $n \cdot \phi < 1$. $i = 2$ is connected to all $j \in J$, while $i = 1$ is connected only to $j = 1$. We can verify that (1) $\overline{\mathbf{PT}}$ and $\overline{\mathbf{PS}}$ each have a* unique *optimal solution and the two are the same, which is $\mathbf{x}^* = (x_{ij})$ with $x_{11} = 1$, $x_{21} = 0$, and $x_{2j} = 1$ for all $1 < j \leq n$; (2) for $\overline{\mathbf{PS}}$: $\rho_1(\mathbf{x}^*) = \phi$, and $\rho_2(\mathbf{x}^*) = (n-1)\phi < 1$; for $\overline{\mathbf{PT}}$: $\bar{w}_1(\mathbf{x}^*) = 1/(1-\rho_1(\mathbf{x}^*)) - 1 = 1/(1-\phi) - 1$ and $\bar{w}_j = 1/(1-\rho_2(\mathbf{x}^*)) - 1 = 1/(1-(n-1)\phi) - 1$ for $1 < j \leq n$.*

We will now present two lemmas that establish together the correctness of Theorem 2.

Lemma 4. $\overline{\eta_t(\mathbf{x}_s^*)} \leq 1/(1-\overline{\eta_s^*}) - 1$

Proof. Since $\mathbf{x}^* = (x_{ij})$ is an optimal solution to $\overline{\mathbf{PS}}$, $\overline{\eta_s^*} = \max_{i \in I} \rho_i(\mathbf{x}^*) := \rho^*$. Observe that for each $j \in J$,

$$\bar{w}_j(\mathbf{x}^*) = \sum_{i \sim j} \frac{x_{ij}}{1-\rho_i(\mathbf{x}^*)} \leq \sum_{i \sim j} \frac{x_{ij}}{1-\rho^*} = \frac{1}{1-\rho^*} - 1,$$

which suggests that $\overline{\eta_t(\mathbf{x}^*)} = \max_j \bar{w}_j(\mathbf{x}^*) \leq 1/(1-\rho^*) - 1 = 1/(1-\overline{\eta_s^*}) - 1$. □

Lemma 5. $1/(1-\overline{\eta_s^*}) - 1 \leq \overline{\eta_t^*}$.

The lemma's proof is in the full version of the paper [Trabelsi *et al.*, 2024].

We're now set to present results for $\kappa = 1$.

Theorem 2. *Consider an input $\boldsymbol{\mu} = (\mu_{ij})$ with $\kappa = 1$. Let $\mathbf{x}_s^*$ be an optimal solution to $\overline{\mathbf{PS}}$. We have that the value of $\overline{\mathbf{PT}}$ on the solution of $\mathbf{x}_s^*$ is equal to its optimal value,* i.e., *$\eta_t(\mathbf{x}_s^*) = \eta_t^*$.*

Proof. The above two lemmas together imply that $\overline{\eta_t(\mathbf{x}_s^*)} \leq \overline{\eta_t^*}$. $\mathbf{x}_s^*$ is feasible to $\overline{\mathbf{PT}}$ since $\overline{\mathbf{PT}}$ and $\overline{\mathbf{PS}}$ share the same set of constraints, and thus, $\overline{\eta_t(\mathbf{x}_s^*)} \geq \overline{\eta_t^*}$, which establishes Theorem 2. □

[4]Infeasibility to either Program $\overline{\mathbf{PT}}$ or $\overline{\mathbf{PS}}$ suggests that no policy can lead to meaningful fairness among tasks (finite max expected waiting time) or among workers (a non-zero ratio of being free).

4 Experiments

4.1 Algorithms and Heuristics

This section presents an algorithm derived from solutions to one of the minimax problems. We also describe a heuristic based on this algorithm, which gives preference to assigning tasks to available workers, thereby enhancing allocation through the effective workload of free workers. In addition, this section introduces two real-time greedy heuristics, which function as baseline methodologies.

Minimax problems based algorithm: We first describe Algorithm 1. This algorithm has offline and online phases. In the offline phase (line 2), a solution to one of the minimax problems is computed. In the online phase (lines 4-7), when a task arrives, the task is assigned to the queue of a worker according to the probabilities computed by the program in the offline phase. This algorithm has two variants: One solves **PT** in the offline phase while the other solves **PS**.

Minimax problems based heuristic: A notable issue with Algorithm 1 is that tasks can wait for a busy worker despite other available workers. This leads to suboptimal performance. To address this, we create a heuristic based on Algorithm 1. Like Algorithm 1, in Algorithm 2, task assignment probabilities are computed offline to mitigate this problem. In the online phase, incoming tasks are assigned to free workers. If multiple workers are free, their precomputed probabilities (from the offline phase) are normalized to sum to 1. A worker is subsequently chosen randomly, guided by these normalized probabilities. If there are no free workers, the tasks are assigned according to their probabilities as in Algorithm 1. Algorithm 2 describes this heuristic. As in Algorithm 1, there are two variants of Algorithm 2: One solves **PT** in the offline phase, while the other solves **PS**.

This method targets reduced waiting times, especially during low-load periods. However, this change might decrease worker workload or waiting times for other tasks, as it deviates from calculated optimal probabilities. In practice, we find that the trade-off for worker and task fairness is reasonable, given the substantial benefits for all tasks' fairness.

Computational complexity of Algorithms 1 and 2 Both algorithms 1 and 2 have **offline** and **online** phases. The offline phase is identical for both algorithms and requires the solution of **PT** or **PS**. Following [Cohen *et al.*, 2021], the runtime for solving the linear program- **PS** can be as low as $O^*(N^{2+1/6}\log(N/\delta))$, where δ is the relative accuracy and $N = |E|$ is the number of edges in the graph G. We leave the complexity of solving **PT** to future work. In any case, the complexity of the offline phase dominates the complexity of the online phase.

The two greedy heuristics Similar to [Ackerman Viden *et al.*, 2023], we devised two greedy heuristics as baselines for comparison. The first minimizes maximum task waiting times, and the second minimizes maximum worker workload. In the first, incoming tasks are assigned to workers with the shortest estimated waiting time, calculated by summing average expected task durations for tasks in the queue. The elapsed time for ongoing tasks is subtracted from their average duration to update estimates. For the second, tasks are assigned

Algorithm 1: An LP-based algorithm for **FAIR-T** and **FAIR-S**.

Offline Phase:
Solve **PT** (**PS**) and let $\{x_{ij}\}$ be an optimal solution.
Online Phase:
for *each task of type* j *that arrive on time* t **do**
  Let Q_i be a queue of worker i
  Choose randomly a worker i following the probabilities in $\{x_{ij}\}$
  Update $Q_i = Q_i \cup \{(j,t)\}$.

Algorithm 2: A heuristic for **FAIR-T** (**FAIR-S**).

Offline Phase:
Solve **PT** (**PS**) and let $\{x_{ij}\}$ be an optimal solution.
Online Phase:
for *each task of type* j *that arrive on time* t **do**
  Let Q_i be a queue of worker i and let F be the subset of free workers on time t
  If $F \neq \emptyset$, randomly choose a free worker i with probability $\{x_{ij}/\sum_{i' \in F} x_{i'j}\}$
  Otherwise choose randomly a worker i following the probabilities in $\{x_{ij}\}$
  Update $Q_i = Q_i \cup \{(j,t)\}$.

to less utilized workers based on current workload upon task arrival. This approach considers executed tasks, using actual durations rather than expected durations. The first heuristic is denoted as GTW (Greedy Task Waiting time) and the second as GWU (Greedy Worker Workload).

4.2 Experimental Settings

We ran experiments on the teleoperation domain. As already mentioned in Section 1, the teleoperation of AVs involves intervention tasks that are assigned to the teleoperators who perform them. We adapted the dataset of [Ackerman Viden *et al.*, 2023] for our two-sided fairness study. More details about the experimental settings and additional experimental results are in the full version of the paper [Trabelsi *et al.*, 2024]. Source code and data for running the experiments are available at [Trabelsi, 2024].

The tasks, their durations and their arrival rates Our study built upon the four task types defined by Viden et al. (2023). Their dataset provided valuable insights into the average duration times for each teleoperator (worker) and task type in a simulation. We explored three distinct approaches to define task duration in our experiments. All approaches involved sampling durations from exponential distributions, but the difference lay in the means of these distributions.

The teleoperators and the tasks they can perform Using the dataset of Viden et al. (2023), we initially had 10 teleoperators (workers) and 4 task types. We form a bipartite graph with 10 teleoperators on one side and 4 task types on the other. The dataset provides average task completion times for each teleoperator-task pair. An edge is established between a teleoperator and a task type if their average time matches or exceeds the task type's median value. Following this process, a teleoperator who consistently performed tasks slower than the median was identified and subsequently excluded from the graph. More experiments on a synthetic dataset in which the numbers of teleoperators and task types are varied can be found in [Trabelsi *et al.*, 2024].

Experimental environment and more settings Each experiment spanned a virtual 4-week period. Due to algorithmic stochasticity, each experiment was repeated 10 times.

4.3 Results and discussion

Effect of changing κ In Figures 1(a,b), we illustrate the performance of various methods across diverse κ values. In Figure 1(a), we measure the maximum task waiting time. We see that the gap between SIM(**PS**) and SIM(**PT**), as well as the gap between SIM-F(**PS**) and SIM-F(**PT**), increase with κ. This aligns with the fact that with higher values of κ, the approximation ratio of **PS**'s solution relative to **PT**'s objective is greater. However, the ratio between the different methods measured in practice is lower than the worst-case theoretical ratio given by Theorem 1 (which is greater than κ^3).

In Figure 1(b), we measure the maximum worker workload. The differences between SIM(**PT**) vs SIM(**PS**) are very small for $\kappa \leq 3$, but they become more significant for $\kappa \in \{4, 5\}$. Surprisingly, there is a different effect with SIM-F(**PT**) and SIM-F(**PS**). SIM-F(**PT**) performs slightly better than SIM-F(**PS**). We conjecture that the initial selection of free workers has a more detrimental effect in SIM-F(**PS**), which integrates two distinctly different methods, in contrast to the relatively similar approaches in SIM-F(**PT**). We also see that for larger values of κ, both SIM(**PT**) and SIM(**PS**) perform worse than for lower values.

Effect of changing the task load Figures 1(c,d) might help the teleoperation center's owner decide whether the current number of workers is sufficient. It is noticeable that in Figure 1(c) there is a significant jump from 120000 to 140000 tasks per day. This means that perhaps the owner should employ more workers in this case. Referring to Figure 2(d) may lead us to similar conclusions. Employing more workers is advisable if individual worker workload is excessively high.

Effect of changing the task balance Figures 1(e,f) represent the performance of the different algorithms when changing the task balance. The left bar represents an even distribution for each task type (0.25). The second bar represents a higher probability for the first type (0.7) and a lower probability for the other types (0.1). The other bars are similarly defined for the other task types.

In Figures 1(e,f), higher arrival distribution of the first task type leads to elevated waiting times and worker workload. Consequently, the teleoperation center's owner could enhance fairness by upskilling operators who are not qualified for the task or hiring new ones proficient in it. Alternatively, training could be provided to expedite task completion. The negligible error bars in all figures show that the error approaches 0 if the experiments are carried out over a sufficiently long period of time, as we have done.

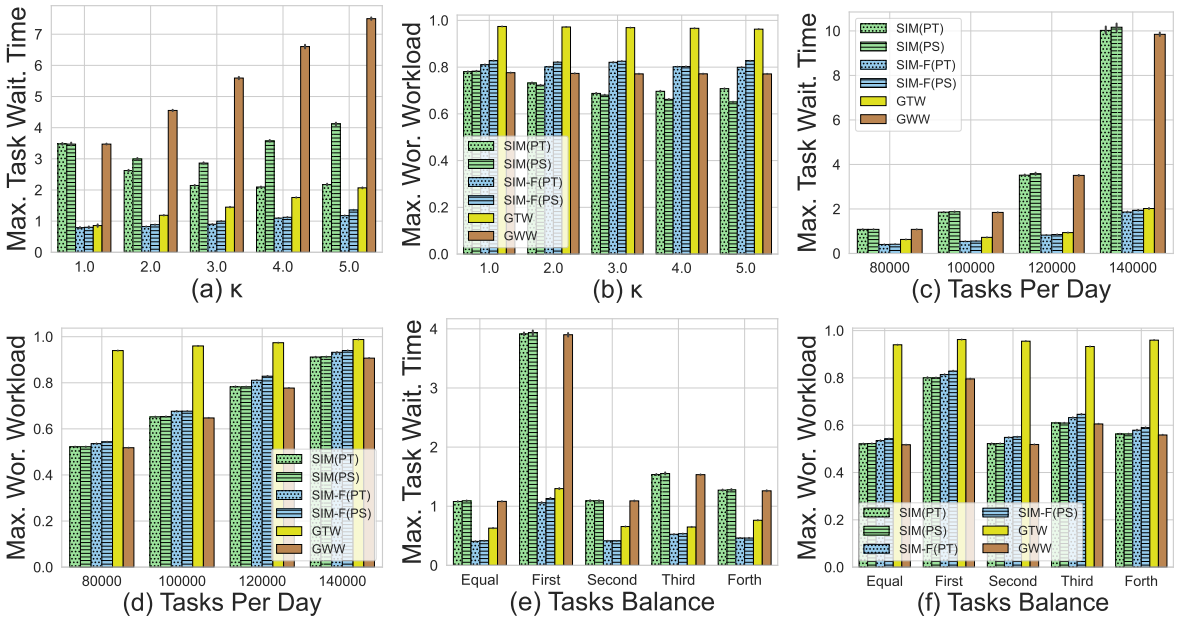

Figure 1: Y axis is the maximum waiting time(in seconds) for a task(a,c and e) and the max. worker workload(b,d and f). The X axis in (a,b) is the value of κ(x axis) and the task load is of 120000 tasks per day. In (c,d), the X axis is the task load and κ is set to 1. In (e,f) the X axis is for different balances of arrival distribution: first bar is for equal distribution for each task type. In the second bar, the first type has probability of 70% and the others have 10%. the other bars are defined similarly for the second, third and forth task types (task load was 80000 tasks per day and κ is set to 1). In the legend: SIM(PT) and SIM(PS) denote Algorithm 1's results for **PT** and **PS** in simulation. SIM-F(PT) and SIM-F(PS) are Algorithm 2's results (in which we assign to a **free** worker first) for **PT** and **PS** in simulation. *GTW* and *GWU* are the results of the greedy heuristics targeting task waiting time and worker workload in simulation. Error bars represent a confidence interval of 0.95.

Computed optimal values vs simulation values In all experiments that we ran, the computed expected maximum waiting time (OPT(**PT**)) and the computed expected maximum worker workload (OPT(**PS**)) closely align with simulation-derived values (SIM(**PT**) and SIM(**PS**) respectively). Additionally, the alignment of OPT(**PT**) and OPT(**PS**) at $\kappa = 1$ is consistent with Theorem 1.

Choosing the best algorithm The heuristic GTW, which minimizes the maximum task waiting time, performs well at maximum task waiting time and performs poorly at maximum worker workload. Conversely, the greedy heuristic that minimizes the maximum worker workload, GWU, performs well at the maximum worker workload and performs poorly at maximum task waiting time. The methods that offer the best tradeoff between two dimensions of fairness are SIM-F(**PT**) and SIM-F(**PS**). However, since **PT** is nonlinear, there is no tool that guarantees to find an optimal solution for **PT**, and therefore **PT**-dependent approaches such as SIM-F(**PT**) might be unsolvable.

Therefore, if $\kappa = 1$ or at least a small number close to 1, we might want to use SIM-F(**PS**). However, SIM(**PS**) might be slightly better if worker workload is more important than task waiting times (but still important). If κ is large, it is advisable to consider using a tool that approximates a solution for **PT** with SIM-F(**PT**). The figures show that the available tools work adequately in such cases, despite the lack of theoretical guarantees (at least for small problems). Another option is to try both SIM-F(**PT**) and SIM-F(**PS**) and pick the one that gives the best results.

5 Conclusion

This paper addresses two-sided fairness problems represented as online bipartite matching with accommodated delays. We introduce two minimax problems: **PT** to minimize the maximum workload of workers and **PS** to minimize the maximum waiting time of tasks. We show that the second problem can be formulated as a linear program and thus solved efficiently. Moreover, we showed that the policy using a solution for **PS** approximates the solution for **PT**, and we then presented an upper bound on the approximation ratio. Finally, we compared the performance of different approaches (most of them used the solutions to the problems) and empirically evaluated their performance.

Future research may explore different definitions of fairness. In addition, it is promising to extend our approach to scenarios where workers are also arriving dynamically. To demonstrate the need in such scenarios one might consider the teleoperation application where teleoperators (workers) can join or leave the crew. Considering different distributions for both task arrivals and task durations can provide more depth and insights into the study. Finally, it might be beneficial to consider some robust version, say, minimization of the maximum possible absolute waiting time among users, which is equivalent to the minimization of the maximum absolute waiting time among all workers.

Acknowledgements

This research has been partially supported by the Israel Science Foundation under grant 1958/20 and the EU Project TAILOR under grant 952215. Work of Pan Xu was partially supported by NSF CRII Award IIS-1948157.

References

[Ackerman Viden *et al.*, 2023] Osnat Ackerman Viden, Yohai Trabelsi, Pan Xu, Karthik Abinav Sankararaman, Oleg Maksimov, and Sarit Kraus. Allocation problem in remote teleoperation: Online matching with offline reusable resources and delayed assignments. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 513–521, 2023.

[Asmussen, 2003] Søren Asmussen. Random walks. *Applied Probability and Queues*, pages 220–243, 2003.

[Cohen *et al.*, 2021] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)*, 68(1):1–39, 2021.

[Delong *et al.*, 2022] Steven Delong, Alireza Farhadi, Rad Niazadeh, and Balasubramanian Sivan. Online bipartite matching with reusable resources. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 962–963, 2022.

[Devore, 2008] Jay L Devore. Probability and statistics for engineering and the sciences. 2008.

[Dickerson *et al.*, 2021] John P Dickerson, Karthik A Sankararaman, Aravind Srinivasan, and Pan Xu. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. *ACM Transactions on Economics and Computation (TEAC)*, 9(3):1–17, 2021.

[Esmaeili *et al.*, 2023] Seyed Esmaeili, Sharmila Duppala, Davidson Cheng, Vedant Nanda, Aravind Srinivasan, and John P Dickerson. Rawlsian fairness in online bipartite matching: Two-sided, group, and individual. In *Proc. 37th AAAI*, number 5, pages 5624–5632, 2023.

[Gallager, 2011] Robert G Gallager. Discrete stochastic processes. *OpenCourseWare: Massachusetts Institute of Technology*, 2011.

[Gupta and Goyal, 1964] SK Gupta and JK Goyal. Queues with poisson input and hyper-exponential output with finite waiting space. *Operations Research*, 12(1):75–81, 1964.

[Karp *et al.*, 1990] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. *STOC-90*, 1990.

[Kendall, 1953] David G Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953.

[Lesmana *et al.*, 2019] Nixie S Lesmana, Xuan Zhang, and Xiaohui Bei. Balancing efficiency and fairness in on-demand ridesourcing. *Advances in neural information processing systems*, 32, 2019.

[Li *et al.*, 2023] Zihao Li, Hao Wang, and Zhenzhen Yan. Fully online matching with stochastic arrivals and departures. In *Proc. 37th AAAI*, number 10, pages 12014–12021, 2023.

[Ma *et al.*, 2020] Will Ma, Pan Xu, and Yifan Xu. Group-level fairness maximization in online bipartite matching. *arXiv preprint arXiv:2011.13908*, 2020.

[Maister and others, 1984] David H Maister et al. *The psychology of waiting lines*. Citeseer, 1984.

[Patro *et al.*, 2020] Gourab K Patro, Arpita Biswas, Niloy Ganguly, Krishna P Gummadi, and Abhijnan Chakraborty. Fairrec: Two-sided fairness for personalized recommendations in two-sided platforms. In *Proceedings of the web conference 2020*, pages 1194–1204, 2020.

[Rawls, 1958] John Rawls. Justice as fairness. *The philosophical review*, 67(2):164–194, 1958.

[Rawls, 1999] John Rawls. *A Theory of Justice*. Harvard University Press, Cambridge, MA, 1999.

[Righter, 1987] Rhonda Righter. The stochastic sequential assignment problem with random deadlines. *Probability in the Engineering and Informational Sciences*, 1(2):189–202, 1987.

[Rigter *et al.*, 2022] Marc Rigter, Danial Dervovic, Parisa Hassanzadeh, Jason Long, Parisa Zehtabi, and Daniele Magazzeni. Optimal admission control for multiclass queues with time-varying arrival rates via state abstraction. In *Proc. 36th AAAI*, number 9, pages 9918–9925, 2022.

[Tener and Lanir, 2022] Felix Tener and Joel Lanir. Driving from a distance: Challenges and guidelines for autonomous vehicle teleoperation interfaces. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2022.

[Trabelsi *et al.*, 2024] Yohai Trabelsi, Pan Xu, and Sarit Kraus. Design a win-win strategy that is fair to both service providers and tasks when rejection is not an option. https://github.com/yohayt/two_sided_fairness/blob/main/full_version.pdf?raw=true, 2024.

[Trabelsi, 2024] Yohai Trabelsi. Code and data: Design a win-win strategy that is fair to both service providers and tasks when rejection is not an option. https://github.com/yohayt/two_sided_fairness, 2024. Accessed: 05/05/2024.

[Zhang, 2020] Tao Zhang. Toward automated vehicle teleoperation: Vision, opportunities, and challenges. *IEEE Internet of Things Journal*, 7(12):11347–11354, 2020.

[Zhou *et al.*, 2023] Quan Zhou, Jakub Mareček, and Robert Shorten. Subgroup fairness in two-sided markets. *Plos one*, 18(2):e0281443, 2023.